IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Brian C. Barnes, et al	Group Art Unit: 2132
Serial No.: 10/005,248	Examiner: Lemma, Samson B.
Filed: December 3, 2001	Atty. Dkt. No.: 2000.056500/TT4085
	Confirmation No.: 7937
For: Method And Apparatus For Restricted Execution Of Security Sensitive Instructions	

AMENDED APPEAL BRIEF

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

On November 22, 2006, Appellants filed a Notice of Appeal in response to a Final Office Action dated August 24, 2006, issued in connection with the above-identified application. In support of their appeal, Appellants submitted an Appeal Brief to the Board of Patent Appeals and Interferences. The Examiner filed a Notification of Non-compliant Appeal Brief on March 7, 2007. Applicants hereby file an Amended Appeal Brief in response to the objections raised by the Examiner in the Notification of Non-compliance. The one-month due date for a response is April 7, 2007. This paper is being filed on or before the due date, therefore, it is timely filed.

If an extension of time is required to enable this paper to be timely filed and there is no separate Petition for Extension of Time filed herewith, this paper is to be construed as also constituting a Petition for Extension of Time Under 37 CFR § 1.136(a) for a period of time sufficient to enable this document to be timely filed.

No fees are believed to be due as a result of filing this paper. However, should any additional fees under 37 C.F.R. §§ 1.16 to 1.21 be required for any reason, the Commissioner is authorized to deduct said fees from Williams, Morgan & Amerson's PTO Account 50-0786/2000.056500.

I. REAL PARTY IN INTEREST

The present application is owned by Advanced Micro Devices, Inc.

II. RELATED APPEALS AND INTERFERENCES

Appellants are not aware of any appeals, interferences, or judicial proceedings that are related to, may be affected by, or might affect or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-24 are pending in the instant application. All the pending claims are the subject of the present appeal and they are attached as Appendix A. Claims 1-3, 7-11, 15-19 and 23-24 stand rejected in the Final Office Action issued on August 24, 2006 under 35 U.S.C. §102(b) as allegedly being anticipated by ***Draves*** (U.S. Patent No. 5,802,590). Claims 4-6, 12-14 and 20-22 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over ***Draves*** in view of ***Krueger*** et al. (U.S. Patent No. 4,962,533). Claims 1-3, 7-11, 15-19 and 23-24 were rejected under 35 U.S.C. § 102(b) as allegedly being anticipated by ***Kamiya*** (U.S. Patent No. 4,949,238). Claims 4-6, 12-14 and 20-22 were rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over ***Kamiya*** in view of ***Krueger***.

IV. STATUS OF AMENDMENTS

Applicants believe that there were no amendments filed subsequent to the Final Office Action.

V. SUMMARY OF CLAIMED SUBJECT MATTER

In general, the present invention provides a method and apparatus for ensuring the secure operation of a computer system. A computer system may restrict the execution of security sensitive instructions. Based on a security access scheme, a processor unit may restrict the execution of some security sensitive instructions that are to be carried out by the processor unit. In this way, based upon predetermined security entries programmed in the processor unit, the processor unit may prohibit certain computing tasks from being executed by it. There are three independent claims at issue in the current appeal: claims 1, 9 and 17.

Independent claim 1 provides a method that includes associating a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by a processor. A request is made to execute at least one of the plurality of instructions or set of instructions by a software code running on the processor. A second security ID associated with the software code running on the processor is obtained. A comparison is made between the second security ID with the first security ID. The requested instruction or set of instructions is executed providing that the second security ID matches the first security ID. By way of example only, at least portions of the invention are described in the Specification at p. 8, line 13 – p. 15, line 14; Figures 3-5B.

Independent claim 9 is generally directed to an apparatus comprising a processor (305) for running code thereon, and for associating a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by the processor (305).

The processor (305) receives a request to execute at least one of the plurality of instructions or set of instructions by the code running thereon, obtains a second security ID associated with the code, compares the second security ID with the first security ID, and executes the requested instruction or set of instructions providing that the second security ID matches the first security ID. By way of example only, at least portions of the invention are described in the Specification at p. 8, line 13 –p. 15, line 14; Figures 3-5B.

Independent claim 17 is generally directed to an article comprising one or more machine-readable storage media including instructions that when executed enable a processor (305) to associate a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by the processor (305), and request to execute at least one of the plurality of instructions or set of instructions by software code running on the processor (305). A second security ID associated with the software code running on the processor (305) is obtained. A comparison is made between the second security ID with the first security ID. The requested instruction or set of instructions is executed providing that the second security ID matches the first security ID. By way of example only, at least portions of the invention are described in the Specification at pages p. 8, line 13 –p. 15, line 14; Figures 3-5B.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Appellants respectfully request that the Board review and overturn the rejections present in this case. The following issues are presented on appeal in this case:

a) Whether claims 1-3, 7-11, 15-19 and 23-24 are anticipated by ***Draves*** under 35 U. S. C. §102(b);

b) Whether claims 4-6, 12-14 and 20-22 are obvious over ***Draves*** in view of ***Krueger*** under 35 U.S.C. §103(a);

c) Whether claims 1-3, 7-11, 15-19 and 23-24 are anticipated by ***Kamiya*** under 35 U.S.C. §102(b); and

d) Whether claims 4-6, 12-14 and 20-22 are obvious over ***Kamiya*** in view of ***Krueger*** under 35 U.S.C. §103(a).

VII. ARGUMENT

Appellants respectfully submit that the Examiner erred in rejecting claims 1-24. Appellants respectfully request that the rejection of claims 1-24 over ***Draves*** be reversed.

A. Claims 1-3, 7-11, 15-19 and 23-24 are not anticipated by Draves

In the Office Action mailed August 24, 2006, claims 1-3, 7-11, 15-19 and 23-24 were finally rejected under 35 U.S.C. § 102(b) as allegedly being anticipated by ***Draves*** (U S Patent 5,802,590). Applicants respectfully disagree and submit that the Examiners' position is flawed for multiple reasons, as set forth below.

As the Board well knows, an anticipating reference by definition must disclose every limitation of the rejected claim in the same relationship to one another as set forth in the claim. *In re Bond*, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). As set forth in M.P.E.P. §2131, a claim is anticipated under 35 U.S.C. §102(b) only if **each and every element as set forth** is found, either expressly or inherently described, in a single prior art reference. Under this legal principle, it is respectfully submitted that all pending claims are in condition for allowance.

To the extent the Examiner relies on principles of inherency in making the anticipation rejections in the Office Action, inherency requires that the asserted proposition necessarily flow from the disclosure. *In re Oelrich*, 212 U.S.P.Q. 323, 326 (C.C.P.A. 1981); *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1463-64 (Bd. Pat. App. & Int. 1990); *Ex parte Skinner*, 2 U.S.P.Q.2d 1788,

1789 (Bd. Pat. App. & Int. 1987); *In re King*, 231 U.S.P.Q. 136, 138 (Fed. Cir. 1986). It is not enough that a reference could have, should have, or would have been used as the claimed invention. "The mere fact that a certain thing may result from a given set of circumstances is not sufficient." *Oelrich*, at 326, quoting *Hansgirg v. Kemmer*, 40 U.S.P.Q. 665, 667 (C.C.P.A. 1939); *In re Rijckaert*, 28 U.S.P.Q.2d 1955, 1957 (Fed. Cir. 1993), quoting *Oelrich*, at 326; see also *Skinner*, at 1789. "Inherency ... may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient." *Skinner*, at 1789, citing *Oelrich*. Where anticipation is found through inherency, the Office's burden of establishing prima facie anticipation includes the burden of providing "...some evidence or scientific reasoning to establish the reasonableness of the examiner's belief that the functional limitation is an inherent characteristic of the prior art." *Skinner* at 1789.

Independent claim 1, among other things, describes and claims a method for associating a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by a processor. The method further comprises requesting to execute at least one of the plurality of instructions or set of instructions by a software code running on the processor, obtaining a second security ID associated with the software code running on the processor, comparing the second security ID with the first security ID; and executing the requested instruction or set of instructions providing that the second security ID matches the first security ID. Independent claims 9 and 17 also set forth, among other things, requesting to execute at least one of the plurality of instructions or set of instructions by the software code running on the processor. Claims 9 and 17 also set forth obtaining a second security ID associated with the software code and executing the requested instruction or set of instructions

providing that the second security ID matches the first security ID. It is respectfully submitted that the Examiner erred in rejecting independent claims 1, 9, and 17.

Claim 1 thus calls for associating a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by a processor. In claim 1, a second security ID associated with the software code running on the processor is obtained for executing the requested instruction or set of instructions providing that the second security ID matches the first security ID. Thus, claim 1 states use of two different security IDs for two distinct entities one of which executes the other. While one of the security IDs is associated with the security sensitive instructions to be executed, the other security ID is obtained for the software code that executes these security sensitive instructions. That is, by using a first security ID, a group of instructions are classified as security sensitive instructions before execution. Indeed, the Specification describes that a process 500 commences at block 505 where an instruction or a set (i.e., grouping) of instructions are classified as security sensitive, and each such instruction or set of instructions is associated with a security ID at the processor 305 at page 12, lines 14-18. However, a second security ID of the software code currently being executed by the processor is obtained. To obtain the second security ID for the software code running on the processor, for example, the address of a virtual memory table that corresponds to the security sensitive instruction or set of instructions that the software code desires to execute may be referenced. For providing that the second security ID matches the first security ID, when a particular security sensitive instruction is requested by the currently executed code on the processor 305, the security ID of the currently executing code obtained by referencing the address of the virtual memory access table 405 is compared by the processor 305 to the security ID for the particular security instruction or set of instructions that the code desires to execute.

See Applicants' Specification at page 12, lines 1-7. The processor 305 determines whether the software code currently running thereon is attempting to execute an instruction that has been classified as security sensitive. See Applicants' Specification at page 12, lines 14-16. In other words, claim 1 calls for providing the process 500 for restricting the execution of security sensitive instructions by code running on the processor 305. Thus, the claims 1, 9, and 17, when read in light of the Specification, indicate that the one of the two different security IDs is obtained for a running process that executes one or more security sensitive classified instructions using another one of the two different security IDs which is distinct from ensuring that only authorized processes are able to access resources disclosed by the ***Draves*** reference.

In particular, the Examiner relies upon the ***Draves*** reference, asserting that all the elements of claims 1, 9 and 17 are taught by ***Draves***. The Applicants respectfully disagree.

Draves describes techniques for granting only authorized processes a secure access to a shared computer system resource. The system described by ***Draves*** ensures that a computer program is authorized to access a computer system resource. In particular, ***Draves*** refers to a process as each concurrently executing computer program. See ***Draves***, column 1, lines 14-15. While each concurrently executing computer program is referred to as a process, various resources include the central processing unit, main memory, and peripheral devices (e.g., disk drives and printers). See ***Draves,*** column 1, lines 14-19. Since processes frequently need to share resources, to help manage the various resources, a kernel maintains a resource table for each process. See ***Draves,*** column 1, lines 23 and lines 42-43. The kernel allows a process access to a resource only when the passed key matches the key for the resource that is stored in the resource entry. See ***Draves***, column 1, lines 16-20.

Draves does not describe or suggest obtaining a second security ID of the software code currently being executed by the processor. By contrast, Applicants' the second security ID for the software code running as described in the Specification is obtained for the software code currently being executed by the processor. See Patent Application, page 13, lines 16-22. To obtain the second security ID for the software code currently being executed on the processor, for example, the address of a virtual memory table that corresponds to the security sensitive instruction or set of instructions that the software code desires to execute may be referenced.

However, ***Draves*** matches the key of a requesting process to the key for a resource to which access is allowed by a process called kernel. In other words, the second security ID associated with the software code running is not obtained. ***Draves*** is completely silent about restricting the execution by the kernel (for which a security ID has been obtained and matched) of the instructions classified as security sensitive based on the first security ID. Instead, in ***Draves*** the server process 302 sends a resource allocation request to the kernel 304 for sharing the resource with the client process 314. The handle/key pairs for the shared resource are passed by the server process 302. For example, when a process wishes to access the allocated resource, it simply passes the handle/key pair associated with a shared computer system resource to the kernel. The kernel examines the resource entry indexed by the passed handle to determine whether the passed key is equal to the key in the indexed resource entry. In this way, through the use of handle/key pairs, ***Draves*** provides a system which ensures that only authorized processes are able to access resources. The kernel allows a process access to a resource only when the passed key matches the key for the resource that is stored in the resource entry. See ***Draves,*** column 3, lines 63-67. ***Draves*** thus fails to obtain the second security ID for the software code running on the processor and does not teach or suggest executing the requested instruction or set

of instructions providing that the second security ID associated with the software code running on the processor matches the first security ID, as set forth in independent claim 1.

The Examiner, on page 5 of the Final Office Action, alleges that the process corresponds to "instructions", as set forth in claims 1, 9 and 17. See ***Draves*** column 3, lines 60-62. The Examiner further alleges a resource identifier comprising the handle/key pair associated with the same process /program/code corresponds to the second security ID associated with the software code. Applicants disagree. It is respectfully submitted that the Examiner's rejection is improper because the Examiner cannot satisfy two claim rejection features with the same element. That is, the Examiner uses the same "process" in ***Draves*** to satisfy the requirement of "instruction" and another distinct requirement of the "software code" that executes the "process," *i.e.*, instruction(s), as set forth in claim 1. The Examiner effectively ignores the teachings of ***Draves*** and the Applicants' Specification. This is clearly improper because it is in direct contravention to the Federal Circuit precedent expressed in Phillips v. AWH, Corp., 415 F.3d 1303 (Fed. Cir. 2005) (*en banc*). Based on the above-indicated standard, it is respectfully submitted that ***Draves*** fails to specifically point out the entirety of all the features set forth in claims 1, 9 and 17. Thus, it is respectfully submitted that each of these independent claims is allowable over the art of record for at least the aforementioned reasons.

At page 3 of the Final Office Action, the Examiner states that ***Draves*** on column 2, lines 27-31, discloses a system for ensuring that a computer program is authorized to access a computer system resource. As noted by the Examiner, on page 4 of the Office Action dated April 19, 2006, on column 3, lines 39-41, the main feature of the ***Draves'*** invention is to provide a secure access to resources. The Examiner further notes that the system generates a system-wide resource table that has a resource entry for each allocated resource. Since each resource

entry contains a preferably non-forgeable key that uniquely identifies the resource, the Examiner further indicates that ***Draves*** on column 3, lines 42-48, discloses that a kernel maintains a system-wide resource table that is a hash table and that contains a resource entry corresponding to each resource allocated by the kernel.

However, as can be seen from above, ***Draves*** at least does not teach requesting to execute at least one of the plurality of instructions or set of instructions by a software code running on the processor. Additionally, ***Draves*** does not teach obtaining a second security ID associated with the software code or comparing such a second security ID with a first security ID of the instructions. Instead of providing a specific citation(s) in the ***Draves*** reference relied upon to make the section 102 rejections, in the Final Office Action, at page 6, the Examiner simply makes a conclusory statement that the second security ID could be provided to a program which is attempting forgery, however, it would not be able to access the requested resources since its security ID/identifier/Key pair would not be the same with the first Security ID which is provided to some other program. In the Final Office Action, at page 6, the Examiner merely appears to suggest that the application programs described by ***Draves*** on column 23-25 (such as word programs and spreadsheet programs) could have a shared memory but one of the program would be able to access the resource of the other program if and only if it has one and the same key pairs/identifier. According to the Examiner, access to the resource of the other program would be otherwise denied as described on column 3, lines 60-column 4, and line 11 in ***Draves***. Since the Examiner fails to provide support for the rejections within the cited reference itself, the Examiner's section 102 rejection of independent claim 1 is flawed. Thus, notwithstanding the Examiner's allegations and suggestions as to the ***Draves*** system, the ***Draves*** reference fails to teach or suggest each claimed feature set forth in claim 1.

In the Advisory Action mailed November 28, 2006, the Examiner on page 2 asserts that ***Draves*** on column 2, lines 27-31, discloses that "the system provides for ensuring that a computer program is authorized to access a computer system resource." According to the Examiner, the system described by ***Draves*** generates a system-wide resource table that has a resource entry for each allocated resource. In ***Draves***, each resource entry contains a preferably non-forgeable key that uniquely identifies the resource. The Examiner then concludes that this indicates the fact that not only key pairs are associated with both a single or same item, i.e., a computer system resource, but may also be associated with several resources such that each of the resources/items are uniquely identified by the non-forgeable keys. The Examiner further asserts that ***Draves*** on column 3, lines 42-48, discloses, "a kernel to maintain a system-wide resource table that is a hash table." While the hash table contains a resource entry corresponding to each resource allocated by the kernel, the allocated resources are identified by a kernel-generated resource identifier. The Examiner alleges that since the ***Draves*** system uses resource identifiers that contain both a handle and a key (a handle/ key pair), it further indicates that several resources/items are identified by the resource identifier or key pair.

On page 2 of the Advisory Action, the Examiner further asserts that ***Draves*** discloses passing the handle/key pair to the kernel when a process wishes to access the allocated resource. The kernel described by ***Draves*** examines the resource entry indexed by the passed handle to determine whether the passed key is equal to the key in the indexed resource entry. The Examiner argues that since the keys may not be equal for several reasons, including resource table compaction and attempted forgery, as disclosed in column 3, line 63 to column 4, line 2 of ***Draves***, the requesting process could be any process including an unauthorized process which is attempting forgery. The Examiner contends that, however, a forgery process is not able to access

other resource that it is not authorized since it does not have the right key pair and the kernel denies this process from accessing the resources by matching the key with the resource it is requesting.

The Examiner further points out that in ***Draves*** when no such resource entry is found, the kernel denies the process access to the resource. On the other hand, the kernel described by ***Draves*** in column 4, lines 7-10 allows the process to access the resource when a resource entry that contains a matching key is found. Finally, the Examiner asserts that since ***Draves*** on column 3, lines 39-41 discloses providing secure access to resources, the ***Draves*** system provides for ensuring that a computer program is authorized to access a computer system resource. That is, according to the Examiner, ***Draves*** controls access to any resources in the computer system by any computer programs.

According to the Examiner and as understood by the Applicants, ***Draves*** discloses that when a process wishes to access the allocated resource, it passes the handle/key pair to the kernel. The kernel then examines the resource entry indexed by the passed handle to determine whether the passed key is equal to the key in the indexed resource entry. The Examiner points out that in ***Draves*** when no such resource entry is found, the kernel denies the process access to the resource. On the other hand, when a resource entry that contains a matching key is found, the kernel allows the process to access the resource." See ***Draves,*** column 4, lines 7-10. In this manner, the Examiner asserts that since ***Draves*** on column 3, lines 39-41 discloses a technique for providing processes a secure access to resources, it ensures that a computer program is authorized to access a particular resource and therefore it teaches all the features recited in claim 1.

Instead of requesting to execute at least one instruction by the software code running on the processor and executing the requested instruction, in ***Draves***, the server process 302 sends a resource allocation request to the kernel 304 for sharing the resource with the client process 314. The handle/key pairs for the shared resource are passed by the server process 302. In ***Draves***, for example, when a process wishes to access the allocated resource, it simply passes the handle/key pair associated with a shared computer system resource to the kernel. The kernel examines the resource entry indexed by the passed handle to determine whether the passed key is equal to the key in the indexed resource entry. In this way, through the use of handle/key pairs, ***Draves*** provides a system which ensures that only authorized processes are able to access resources. The kernel allows a process access to a resource only when the passed key matches the key for the resource that is stored in the resource entry. See ***Draves,*** column 3, lines 63-67.

Draves does not, however, teach or suggest requesting to execute at least one of the plurality of instructions or set of instructions by the software code running on the processor, obtaining a second security ID associated with the software code and executing the requested instruction or set of instructions providing that the second security ID matches the first security ID.

The Examiner further asserts that ***Draves*** discloses an apparatus, comprising a processor for running code thereon, at column 3, lines 39-42 and column 1, lines 11-22 and Figure 2. In ***Draves***, the Examiner alleges that a kernel of an operating system inherently operates in the processor for providing secure access. On column 1, lines 39-42, in ***Draves*** the portion of the operating system that is responsible for the allocation and de-allocation of resources is referred to as the kernel. The kernel interacts with a shell and other programs as well as with the hardware devices on the system. The Examiner alleges that ***Draves*** further discloses, on column 3, lines

60-62, each process that is being defined as concurrently executing computer programs on column 1, lines 14-15. The Examiner states that each process of ***Draves*** corresponds to each of a plurality of instructions or a set of instructions set forth in claim 1. Furthermore, the Examiner states that ***Draves*** on column 3, lines 43-50 discloses that the kernel maintains a system-wide resource table that is a hash table and it contains a resource entry corresponding to each resource allocated by the kernel. Accordingly, the Examiner concludes that each and every limitation of the independent claims 1, 9, and 17 is disclosed by the ***Draves*** reference.

To the contrary, claim 1 describes and sets forth requesting to execute at least one of the plurality of instructions or set of instructions by a software code running on the processor and executing the requested instruction or set of instructions providing that the second security ID matches the first security ID. The Examiner, unfortunately, disregards an express teaching of ***Draves*** and removes any distinction between "process" and "resource" terms to make an anticipation rejection. In particular, the Examiner argues that the "process" in ***Draves*** is a "resource." But equating a "process" to a "resource" is inconsistent with ***Draves***, which does not use these terms interchangeably. That is, none of the various resources described by ***Draves*** are either the plurality of instructions or set of instructions or the software code. In other words, none of the various resources described by ***Draves*** include at least one of the plurality of instructions or set of instructions that have been requested to execute by the software code, as set forth in independent claims 1, 9, and 17. As noted above, ***Draves*** describes that a process wishes to access the allocated resource, which is distinct from the process. The Examiner, however, obfuscates this distinction and collapses the two terms into one. Thus, Applicants respectfully submit that claim 1 is not anticipated by ***Draves*** and request that the Examiner's rejections of claim 1 be withdrawn.

For at least the aforementioned reasons, Applicants respectfully submit that the present invention is not anticipated by ***Draves*** and request that the Examiner's rejections of claim 1 and its dependent claims under 35 U.S.C. § 102(b) be reversed. For at least the aforementioned reasons, the rejections of claims 9-11, 15-19 and 23-24 under 35 U.S.C. 102(b) be reversed.

B. Claims 4-6, 12-14 and 20-22 are not obvious over Draves in view of Krueger

Claims 4-6, 12-14 and 20-22 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over ***Draves*** in view of ***Krueger***.

As the Board well knows, to establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on Applicants' disclosure. *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991); M.P.E.P. § 2142. Moreover, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (CCPA 1974). If an independent claim is nonobvious under 35 U.S.C. § 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596 (Fed. Cir. 1988); M.P.E.P. § 2143.03.

With respect to alleged obviousness, there must be something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination. *Panduit Corp. v. Dennison Mfg. Co.*, 810 F.2d 1561 (Fed. Cir. 1986). In fact, the absence of a suggestion to combine is dispositive in an obviousness determination. *Gambro Lundia AB v. Baxter Health-*

care Corp., 110 F.3d 1573 (Fed. Cir. 1997). The mere fact that the prior art can be combined or modified does not make the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1990); M.P.E.P. § 2143.01. The consistent criterion for determining obviousness is whether the prior art would have suggested to one of ordinary skill in the art that the process should be carried out and would have a reasonable likelihood of success, viewed in the light of the prior art. Both the suggestion and the expectation of success must be founded in the prior art, not in the Applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991; *In re O'Farrell*, 853 F.2d 894 (Fed. Cir. 1988); M.P.E.P. § 2142.

While the Examiner recognizes that ***Draves*** fails to teach or suggest classifying at least one instruction or set of instructions from a plurality of instructions that are to be executed by a processor as being security sensitive, the Examiner relies upon ***Krueger*** to describe these claim limitations. However, ***Krueger*** does not remedy the fundamental deficiencies of ***Draves*** discussed above. Therefore, Applicants submit that claims 4-6, 12-14 and 20-22 are not rendered obvious over ***Draves*** in view of ***Krueger***.

The cited references also fail to provide any suggestion or motivation for modifying the prior art to arrive at Applicants' claimed invention. To the contrary, ***Krueger*** teaches away from classifying instructions as being security sensitive. For example, in column 2, lines 47-48 and lines 53-56, ***Krueger*** does not check classification of an instruction accessing a word in the memory. Instead, ***Krueger*** is directed to controlling user access to data within a computer system. The computer system classifies data (not an instruction or instructions(s)) only at the level which is needed to provide a security technique for a computer system in which all data retains its classification, and in which no data is overclassified. In a computer system every

word in the memory has a corresponding label. This label indicates the security classification, and compartments if any, of that word of data. Each time a word is accessed by any instruction, its classification is checked to see if access is allowed. Any attempt to improperly access any word within the computer system's memory generates a security violation and prohibits further execution of the currently running process. See ***Krueger***, column 2, lines l. 33-56. It is by now well established that teaching away by the prior art constitutes *prima facie* evidence that the claimed invention is not obvious. *See, inter alia, In re Fine,* 5 U.S.P.Q.2d (BNA) 1596, 1599 (Fed. Cir. 1988); *In re Nielson*, 2 U.S.P.Q.2d (BNA) 1525, 1528 (Fed. Cir. 1987); *In re Hedges*, 228 U.S.P.Q. (BNA) 685, 687 (Fed. Cir. 1986).

For at least the aforementioned reasons, Applicants respectfully submit that the present invention is not obvious over the cited references, either alone or in combination. Applicants request that the Examiner's rejections of claims 4-6, 12-14 and 20-22 under 35 U.S.C. 103(a) be reversed.

C. Claims 1-3, 7-11, 15-19 and 23-24 are not anticipated by Kamiya

Claims 1-3, 7-11, 15-19 and 23-24 were rejected under 35 U.S.C. § 102(b) as allegedly being anticipated by ***Kamiya***. The Applicants respectfully disagree.

Kamiya describes an apparatus for detecting memory protection violations in microprogram controlled data processors. To detect a memory protection violation in a data processor for executing microinstructions under control of microprograms, the apparatus comprises privilege level register means for storing a privilege level of a program now being executed. In particular, the data processor comprises a memory protection violation detector 15 and a current privilege level register (CPL) 17 to store the privilege level of a program now being executed are connected. See ***Kamiya***, column 3, lines l. 25-27. The memory protection

violation detector 15 checks whether the memory protection information stored in the attribute information register 16 is correct or false, on the basis of the memory protection branch microinstruction stored in the mask register 122 of the microinstruction register 12 and the privilege level value stored in the current privilege level register 17, in order to detect a memory protection violation. See ***Kamiya***, column 3, lines l. 35-42. However, ***Kamiya*** is completely silent with regard to requesting to execute at least one instruction by the software code running on the processor and executing the requested instruction. Accordingly, ***Kamiya*** fails to teach or suggest a first security identification (ID) being associated with each of the requested instruction(s) to be executed by a software code with which a second security ID is being associated for restricting the execution of the requested instruction(s) by the software code. ***Kamiya*** also fails to teach or suggest obtaining the second security ID associated with the software code that is requested to execute at least one instruction with which the first security ID is being associated, as set forth in claim 1.

For at least the aforementioned reasons, Applicants respectfully submit that the present invention is not anticipated by ***Kamiya*** and request that the Examiner's rejections of claims 1-3, 7-11, 15-19 and 23-24 under 35 U.S.C. 102(b) be reversed.

D. Claims 4-6, 12-14 and 20-22 are not obvious over Kamiya in view of Krueger

Claims 4-6, 12-14 and 20-22 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over ***Kamiya*** in view of ***Krueger***. It is respectfully submitted that the pending claims would not have been obvious in view of the prior art of record.

As the Board well knows, to establish a *prima facie* case of obviousness, three basic criteria must be met. First, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (CCPA 1974).

Second, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. That is, there must be something in the prior art as a whole to suggest the desirability, and thus the obviousness, of making the combination. *Panduit Corp. v. Dennison Mfg. Co.*, 810 F.2d 1561 (Fed. Cir. 1986). In fact, the absence of a suggestion to combine is dispositive in an obviousness determination. *Gambro Lundia AB v. Baxter Healthcare Corp.*, 110 F.3d 1573 (Fed. Cir. 1997). The mere fact that the prior art can be combined or modified does not make the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1990); M.P.E.P. § 2143.01. Third, there must be a reasonable expectation of success.

The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on Applicants' disclosure. *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991); M.P.E.P. § 2142. A recent Federal Circuit case emphasizes that, in an obviousness situation, the prior art must disclose each and every element of the claimed invention, and that any motivation to combine or modify the prior art must be based upon a suggestion in the prior art. *In re Lee*, 61 U.S.P.Q.2d 143 (Fed. Cir. 2002). Conclusory statements regarding common knowledge and common sense are insufficient to support a finding of obviousness. *Id.* at 1434-35. Moreover, it is the claimed invention, as a whole, that must be considered for purposes of determining obviousness. A mere selection of various bits and pieces of the claimed invention from various sources of prior art does not render a claimed invention obvious, unless there is a suggestion or motivation in the prior art for the claimed invention, when considered as a whole.

As discussed above, ***Kamiya*** fails to teach or suggest a first security identification (ID) being associated with each of the requested instruction(s) to be executed by a software code with which a second security ID is being associated for restricting the execution of the requested instruction(s) by the software code. ***Kamiya*** also fails to teach or suggest obtaining the second security ID associated with the software code that is requested to execute at least one instruction with which the first security ID is being associated.

The Examiner relies on ***Krueger*** to further describe the first security ID. The Examiner relies upon ***Krueger*** to describe associating a first security ID comprises classifying at least one instruction or set of instructions from a plurality of instructions that are to be executed by a processor as being security sensitive. However, ***Krueger*** is completely silent with regard to classification of an instruction accessing a word in the memory. Instead, to control user access to data within a computer system, the ***Krueger*** computer system classifies data at the level which is needed to provide a security technique. Consequently, ***Krueger*** does not describe or suggest classifying at least one instruction or set of instructions from a plurality of instructions that are to be executed by a processor as being security sensitive.

It is respectfully submitted that since ***Draves*** fails to anticipate independent claims 1, 9 and 17, for at least the reasons set forth above, ***Draves*** also fails to anticipate their respective dependent claims 1-3 and 7-8, 10-11 and 15-16, 18-19 and 23-24. Applicants respectfully request that the Examiner's rejections of claims 1-3, 7-11, 15-19 and 23-24 under 35 U.S.C. §102(b) and of claims 4-6, 12-14 and 20-22 under § 103(a) be reversed.

VIII. CLAIMS APPENDIX

The claims that are the subject of the present appeal (claims 1-24) are set forth in the attached Claims Appendix.

IX. EVIDENCE APPENDIX

There is no separate Evidence Appendix for this appeal.

X. RELATING PROCEEDINGS APPENDIX

There is no Related Proceedings Appendix for this appeal.

In view of the foregoing, Appellants respectfully submit that the Examiner's assertions that the inventions defined in claims 1-24 are not allowable over the cited prior art are misplaced. It is respectfully submitted that the Examiner erred in not allowing all claims pending in the present application over the prior art of record. That is, Appellants respectfully submit that ***Draves*** does not disclose the entirety of the instant invention set forth in independent claims 1, 9 and 17 or the claims depending therefrom. Accordingly, Appellants respectfully request that the Board review and overturn the §102 rejections present in this case. Appellants respectfully further request the rejections of claims 4-6, 12-14 and 20-22 over ***Draves*** and ***Krueger*** be reversed because the prior art does not teach or suggest any of the pending claims. Appellants respectfully further request the rejections of claims 1-3, 7-11, 15-19 and 23-24 over ***Kamiya*** be reversed because the prior art does not teach or suggest any of the pending claims. Appellants respectfully further request the rejections of claims 4-6, 12-14 and 20-22 over ***Kamiya*** and ***Krueger*** be reversed because the prior art does not teach or suggest any of the pending claims.

For at least the aforementioned reasons, Appellants respectfully request the Board reverse the Examiner's rejections of all the pending claims. The undersigned agent may be contacted at (713) 934-4089 with respect to any questions, comments or suggestions relating to this appeal.

Please date stamp and return the enclosed postcard to evidence receipt of this document.

Respectfully submitted,

WILLIAMS, MORGAN & AMERSON

Date: April 5, 2007

/Jaison C. John/

Jaison C. John
Reg. No. 50,737
10333 Richmond, Suite 1100
Houston, Texas 77042
Telephone: (713) 934-4069
Facsimile: (713) 934-7011
AGENT FOR APPLICANTS

CLAIMS APPENDIX

1. (Original) A method, comprising:

associating a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by a processor;

requesting to execute at least one of the plurality of instructions or set of instructions by a software code running on the processor;

obtaining a second security ID associated with the software code running on the processor;

comparing the second security ID with the first security ID; and

executing the requested instruction or set of instructions providing that the second security ID matches the first security ID.

2. (Original) The method of claim 1, further comprising:

denying the execution of the requested instruction or set of instructions providing that the first and second security IDs mismatch.

3. (Original) The method of claim 1, wherein associating a first security identification (ID) further comprises:

storing a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by a processor.

4. (Original) The method of claim 1, wherein associating a first security identification (ID) further comprises:

classifying at least one instruction or set of instructions from a plurality of instructions that are to be executed by a processor as being security sensitive; and

associating a first security identification (ID) with each of the instructions or set of instructions that are classified as security sensitive.

5. (Original) The method of claim 4, wherein obtaining a second security ID associated with the software code running on the processor, further comprises:

determining whether the requested instruction is classified as security sensitive; and

obtaining a second security ID associated with the software code running on the processor providing the requested instruction was determined to be security sensitive.

6. (Original) The method of claim 5, wherein determining whether the requested instruction is classified as security sensitive, further comprises:

determining whether the requested instruction has the first security ID that is stored.

7. (Original) The method of claim 1, wherein comparing the second security ID with the first security ID further comprises:

comparing a portion of the second security ID with a portion of the first security ID.

8. (Original) The method of claim 7, wherein executing the requested instruction or set of instructions providing that the second security ID matches the first security ID further comprises:

executing the requested instruction or set of instructions providing that the portion of the second security ID matches the portion of the first security ID.

9. (Original) An apparatus, comprising:

a processor for running code thereon, and for associating a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by the processor;

wherein the processor receives a request to execute at least one of the plurality of instructions or set of instructions by the code running thereon, obtains a second security ID associated with the code, compares the second security ID with the first security ID, and executes the requested instruction or set of instructions providing that the second security ID matches the first security ID.

10. (Original) The apparatus of claim 9, wherein the processor denys the execution of the requested instruction or set of instructions providing that the first and second security IDs mismatch.

11. (Original) The apparatus of claim 9, wherein the processor stores the first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by a processor within a programmable register.

12. (Original) The apparatus of claim 9, wherein the processor classifies at least one instruction or set of instructions from a plurality of instructions that are to be executed as being

security sensitive, and associates the first security identification (ID) with each of the instructions or set of instructions that are classified as security sensitive.

13. (Original) The apparatus of claim 12, wherein the processor determines whether the requested instruction is classified as security sensitive, and obtains the second security ID associated with the software code running on the processor providing the requested instruction was determined to be security sensitive.

14. (Original) The apparatus of claim 13, wherein the processor determines whether the requested instruction has the first security ID stored therewith within the programmable register.

15. (Original) The apparatus of claim 9, wherein the processor compares at least a portion of the second security ID with at least a portion of the first security ID.

16. (Original) The apparatus of claim 15, wherein the processor executes the requested instruction or set of instructions providing that the portion of the second security ID matches the portion of the first security ID.

17. (Original) An article comprising one or more machine-readable storage media including instructions that when executed enable a processor to perform:

associating a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by the processor;

requesting to execute at least one of the plurality of instructions or set of instructions by a software code running on the processor;

obtaining a second security ID associated with the software code running on the processor;

comparing the second security ID with the first security ID; and

executing the requested instruction or set of instructions providing that the second security ID matches the first security ID.

18. (Original) The article of claim 17, further comprising:

denying the execution of the requested instruction or set of instructions providing that the first and second security IDs mismatch.

19. (Original) The article of claim 17, wherein associating a first security identification (ID) further comprises:

storing a first security identification (ID) with each of a plurality of instructions or a set of instructions that are to be executed by a processor.

20. (Original) The article of claim 17, wherein associating a first security identification (ID) further comprises:

classifying at least one instruction or set of instructions from a plurality of instructions that are to be executed by a processor as being security sensitive; and

associating a first security identification (ID) with each of the instructions or set of instructions that are classified as security sensitive.

21. (Original) The article of claim 20, wherein obtaining a second security ID associated with the software code running on the processor, further comprises:

determining whether the requested instruction is classified as security sensitive; and

obtaining a second security ID associated with the software code running on the processor providing the requested instruction was determined to be security sensitive.

22. (Original) The article of claim 21, wherein determining whether the requested instruction is classified as security sensitive, further comprises:

determining whether the requested instruction has the first security ID that is stored.

23. (Original) The article of claim 17, wherein comparing the second security ID with the first security ID further comprises:

comparing a portion of the second security ID with a portion of the first security ID.

24. (Original) The article of claim 23, wherein executing the requested instruction or set of instructions providing that the second security ID matches the first security ID further comprises:

executing the requested instruction or set of instructions providing that the portion of the second security ID matches the portion of the first security ID.